الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels
Centre National de l'Enseignement Professionnel à Distance (CNEPD)
المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS DE STRUCTURE MACHINE

SÉRIE 05

OBJECTIF PÉDAGOGIQUE

À la fin de ce cours le stagiaire doit être capable de **c**onnaître :

1- Les différents modes d'adressage

2- La structure des instructions machines

PLAN DE LA LEÇON :

INTRODUCTION :

La mémoire centrale de l'ordinateur est constituée d'un ensemble ordonné de 2^m cellules, chaque cellule contenant un mot de n bits. Ces mots permettent de conserver indifféremment programmes et données. On accède à n'importe laquelle de ces cellules au moyen de son *adresse*, nombre entier compris dans l'intervalle $[0,2m-1]$. Cet accès est en temps constant quelle que soit la valeur de l'adresse.

I- LES TYPES D'ADRESSAGE :

Dans la réalité ce n'est pas tout à fait aussi simple. Une instruction n'est pas en général contenue sur un seul mot mémoire. Les adresses circulant sur le bus d'adresses ne sont pas toujours les adresses réelles des informations dans la mémoire. On rencontre ainsi diverses techniques permettant de retrouver l'adresse physique de l'information (instruction ou donnée) dans la mémoire, techniques regroupées sous l'appellation de **types d'adressage**.

Ces divers modes d'adressage, qui sont en principes transparents pour le programmeur tant qu'il ne travaille pas en langage d'assemblage (langage machine), ont pour but de lui rendre plus facile l'emploi des données. D'une manière générale on peut distinguer 7 modes d'adressage :

- Adressage immédiat,
- Adressage direct (absolu),
- Adressage indirect,
- Adressage relatif,
- Adressage indexé

1- Adressage immédiat :

Il ne s'agit pas d'un adressage à proprement parler, dans la mesure où la partie adresse de l'instruction ne contient pas d'adresse de l'opérande mais bel et bien l'opérande lui-même.

Exemple : ADD A, 1B : additionne la valeur 1B à la valeur contenue dans l'accumulateur. Ceci est différent de l'addition entre la valeur contenue à l'adresse 1B et la valeur contenue dans l'accumulateur.

2- Adressage absolu (adressage direct) :

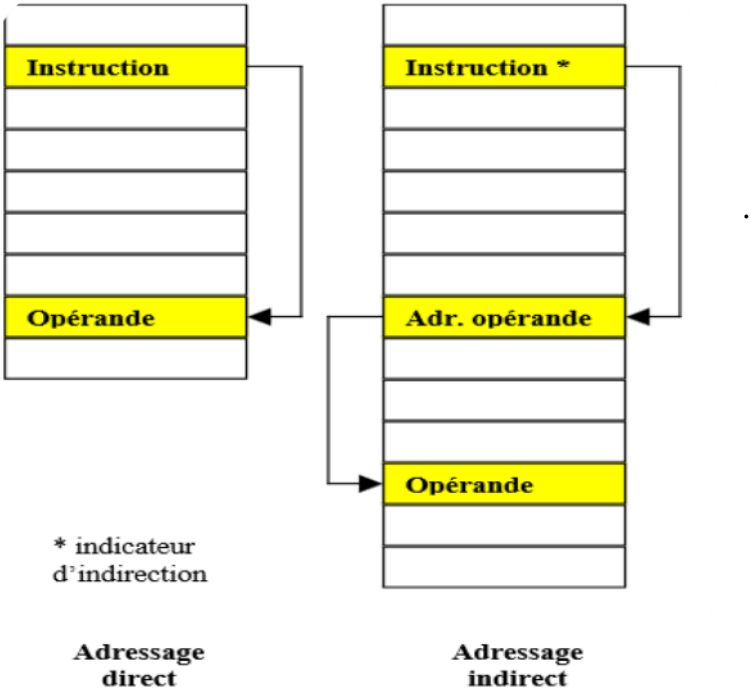
L'adressage est dit absolu, quand le code opération est suivi de l'adresse réelle physique de l'opérande sur lequel travaille l'instruction.

Exemple : LD (F800), A : cette instruction va ranger le contenu du registre accumulateur à l'adresse absolue F800.

3- Adressage indirect :

Un adressage est dit indirect s'il permet d'accéder, non pas à l'information recherchée, mais à un mot mémoire dans lequel on trouvera l'adresse effective de l'information. Ce type d'adressage est assez utile dans la mesure où le code généré tient en règle générale sur un seul octet.

Exemple : ADD A, (HL) : cette instruction va ajouter au contenu de l'accumulateur la donnée de trouvant à l'adresse citée dans le registre HL. Ainsi, si le registre HL contient la valeur F800, on ajoutera à A la donnée contenue à l'adresse F800.



4- Adressage relatif :

Une adresse relative n'indique pas en valeur absolue l'emplacement de l'information en mémoire, mais la situe par rapport à une adresse de référence (déplacement ou offset). Cette adresse de référence étant normalement contenue par le registre PC (Compteur Ordinal).

L'avantage principal de ce mode d'adressage est qu'il permet des branchements efficaces en utilisant les instructions qui tiennent sur deux mots seulement (un mot pour le code opération et un mot pour la référence à l'adresse) ; en fait la zone opérande ne contient pas une adresse mais un déplacement relatif à l'adresse de référence.

Ainsi une instruction avec adressage relatif va permettre un adressage en avant ou en arrière dans la mémoire, ceci par rapport au contenu du compteur ordinal. Compte tenu de la taille accordée au déplacement, celui-ci ne pourra concerner qu'une partie de la mémoire. Ainsi avec un déplacement codé sur 8 bits, on ne pourra adresser qu'une zone de 255 adresses mémoires situées de part et d'autre du contenu courant du compteur ordinal.

Exemple : JR NC, 025 : cette instruction va provoquer un saut en avant de 37 emplacements mémoire si la condition No Carry (pas de retenue) est réalisée.

On peut noter deux avantages à ce type d'adressage :

- D'une part l'amélioration des performances du programme (moins d'octets utilisés),
- D'autre part la possibilité d'implanter le programme n'importe où en mémoire puisque l'on ne considère que des déplacements par rapport au contenu du PC et non pas des adresses absolues.

5- Adressage indexé :

Dans la technique de l'adressage indexé, la valeur spécifiée dans la zone adresse de l'instruction est encore un déplacement mais cette fois ci non pas par rapport au compteur ordinal, mais par rapport au contenu d'un registre spécialisé : le registre d'index.

Exemple : ADD A, (IX+4) : cette instruction va ajouter au contenu de l'accumulateur la donnée se trouvant à l'adresse fournie par le registre IX, augmenté d'un déplacement de 4 octets. Ainsi si le registre IX contient la valeur F800, on prendra la donnée se trouvant à l'adresse F800+4 soit F804.

Considérons un bloc de *n* mots situés aux adresses A, ..., A_n et que l'on veut le déplacer aux adresses B, ..., B_n.

Utilisons l'instruction MOVE A, B qu'il faudrait incrémenter de 1 à *n*. => lourdeur.

C'est pourquoi, on peut utiliser ce mode d'adressage avec MOVE A, IX+k avec IX qui contient l'adresse B et k que l'on peut incrémenter de 1 à n.

II- STRUCTURE D'INSTRUCTIONS :

Les instructions du langage d'assemblage sont représentées par des mnémoniques: une mnémonique est une abréviation du mot représentant l'opération correspondant à l'instruction (**add** pour une addition par exemple). Les registres sont également représentés par des symboles, en général leurs noms (R0, R1, R2 dans la suite). Les valeurs littérales sont représentées directement en décimal (10d) ou encore en hexadécimal (10h ou 0x10). L'utilisation des crochets [] signifie "le contenu de", ainsi [R1] désigne la valeur contenue dans la cellule mémoire dont l'adresse est dans le registre R1 et [1000h] la valeur contenue dans la cellule mémoire d'adresse 1000h.

Le format des instructions décrit l'ensemble des bits des instructions machine telles qu'ils apparaissent en mémoire. Cet ensemble est communément divisé en différents *champs* de longueurs prédéterminés:

Format d'une instruction avec 2 champs adresse			
code opération	modes d'adressage	adresse 1	adresse 2

- **Le *code opération*** spécifie l'opération à effectuer. Il y a une bijection entre ce code et la mnémonique. C'est également ce code qui est utilisé par la micro-architecture pour déterminer l'adresse dans le microprogramme du début de la suite de micro-instructions à exécuter pour réaliser l'instruction machine.
- **Le *mode d'adressage*** spécifie le moyen d'accéder aux opérandes de l'instruction à partir des informations contenues dans les champs adresses. Dans l'exemple ci-dessus, ce champ spécifie que le premier opérande est un registre et le deuxième opérande une adresse absolue (l'adresse en mémoire centrale de l'opérande est présente dans l'instruction, l'opérande est en mémoire).
- **Un *champ adresse*** spécifie l'adresse d'un opérande. La valeur contenue dans le champ adresse peut être **:**

  a) Une valeur *immédiate* (l'opérande est présente directement dans l'instruction): mov R1, 10h
  b) Un numéro de registre: mov R1, R2
  une adresse mémoire: mov R1, [10h]

La valeur de l'opérande dépend du mode d'adressage. Le nombre de champs adresse dépend de l'organisation des registres dans le processeur et du chemin de données.

1- Instruction a une adresse :

Le processeur utilise un *accumulateur*. L'accumulateur Acc est un registre spécial qui est opérande et destination de toutes les opérations arithmétiques et logiques. Deux instructions spécifiques sont ajoutées pour accéder à l'accumulateur: load X pour charger l'accumulateur avec une donnée X et store X pour sauver le contenu de l'accumulateur en mémoire ou dans un autre registre. Toutes les instructions pour les opérations dyadiques n'auront qu'un seul champ adresse, celui du second opérande.

1 champ adresse: source2. add R1 qui calcule Acc <- Acc + R1

2- Instruction a deux ou trois adresses :

- Le processeur dispose de registres généraux. On trouvera dans ce cas des instructions avec :
  a) **2 champs adresses**: une des opérandes reçoit le résultat. add R1, R2 qui calcule R1 <- R1 + R2
  b) **3 champs adresses**: destination, source1, source2. add R1, R2, R3 qui calcule R1 <- R2 + R3

3- Exemple d'instruction :

Traduction d'une instruction d'affectation avec évaluation d'une expression .Soit à traduire l'instruction d'affectation **Python x= (a+b)*(c+d)**, où a, b, c et d sont des variables préalablement définies. Le langage d'assemblage dispose des instructions **add** pour l'addition, **mul** pour la multiplication, **mov** pour le transfert de registre à registre ou de registre à mémoire, et des instructions **load** et **store** ou push et pop suivant le cas. On notera respectivement A, B, C et D les adresses des variables a, b, c et d (ces adresses doivent être déterminées par le programmeur). Ecrivons les programmes correspondants aux différentes situations énumérées ci-dessus.

instructions à 3 adresses	instructions à 2 adresses	accumulateur (1 adresse)
		load A
	mov R1, A	add B
add *R*1, A, B	add R1, B	store T
add R2, C, D	mov R2, C	load C
mul X, R1, R2	add R2, D	add D
	mul R1, R2	mul T
	mov X, R1	store X
		T: temporaire

EXERCICES D'APPLICATION :

EXERCICE N°01 : Soit le programme suivant :

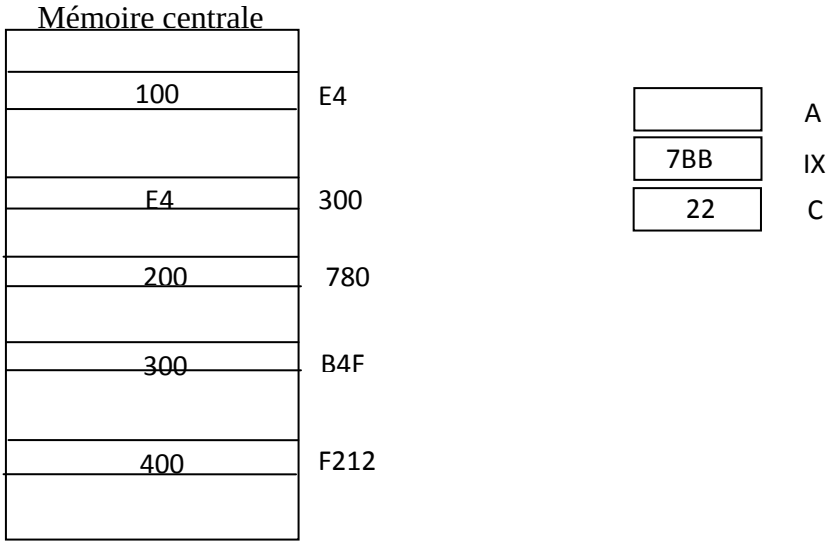
@ D532: LD A, F800

@ 753: ADD A, F810

@ C22 : LD F820, A

1- Quel mode d'adressage utilise-t-il ?
2- Réécrire ce programme en utilisant l'adressage relatif.
3- Réécrire ce programme en utilisant l'adressage indexé : le registre IX contient la valeur B31.

EXERCICE N°02 : Soit la mémoire centrale et les registres représentés comme suit **:**



Que font les 5 programmes suivants :

- LD A, 200 (adressage immédiat)
  ADD A, B4F (adressage direct)
- LD A, 300 (adressage indirect)
  ADD A, F212 (adressage direct)
- @ 760 LD A, 20 (adressage relatif)
  @ 150 ADD A, 150 (adressage relatif)
- LD A, C (adressage par registre)
  ADD A, C (adressage immédiat)
- @ 800 LD A, IX-3B (adressage indexé)
  @ 10 ADD A, F202 (adressage relatif)

EXERCICE N°03

1. Écrire un programme qui lit une entrée de l'utilisateur (chiffre), le stocke dans l'adresse mémoire 10.

2. Quel est le rôle de l'instruction counter?

3. Modifiez votre programme pour que la valeur soit stockée dans une adresse mémoire où se trouve une instruction de votre programme. Que se passe-t-il?

CORRECTIONS DES EXERCICES :

EXERCICE N°01

1. On ne peut pas répondre étant donné que F800, F810 et F820 peuvent être soit des valeurs de données (adressage immédiat) soit des adresses directes (adressage direct), indirectes (adressage indirect) ou des déplacements par rapport à l'adresse de référence contenue dans le compteur ordinal (adressage relatif). De plus, ces différents types d'adressage peuvent être mélangés au sein du même programme.

2- (@ D532): LD A, 22CE

(@ 753): ADD A, F0BD

(@ C22): LD EBFE, A

3- (@ D532): LD A, ECCF

(@ 753): ADD A, ECDF

(@ C22): LD ECEF, A

EXERCICE N°02

* LD A, 200 (adressage immédiat) A = 200
  ADD A, B4F (adressage direct) A = 200 + 300 = 500

* LD A, 300 (adressage indirect) A = 100
  ADD A, F212 (adressage direct) A = 100 + 400 = 500

* @ 760 LD A, 20 (adressage relatif) A = 200
  @ 150 ADD A, 150 (adressage relatif) A = 200 + E4 = 2E4

* LD A, C (adressage par registre) A = 22
  ADD A, C (adressage immédiat) A = 22 + C = 2E

* @ 800 LD A, IX-3B (adressage indexé) A = 200
  @ 10 ADD A, F202 (adressage relatif) A = 200 + 400 = 600

EXERCICE N°03

2. READ
   STORE 10
   STOP
3. L'adresse de l'instruction courante
4. READ
   STORE 0
   STOP
   le simulateur râle (il protège les instructions contre l'écriture)